Statistical Methods for Data Science

Mini Project #1

Yagna Srinivasa Harsha Annadata

Yxa210024

Given:

$f_T(t) = 0.2 \exp(-0.1t) - 0.2 \exp(-0.2t), 0 \le t < \infty,$

0, otherwise,

and $E(T) = 15$ years.

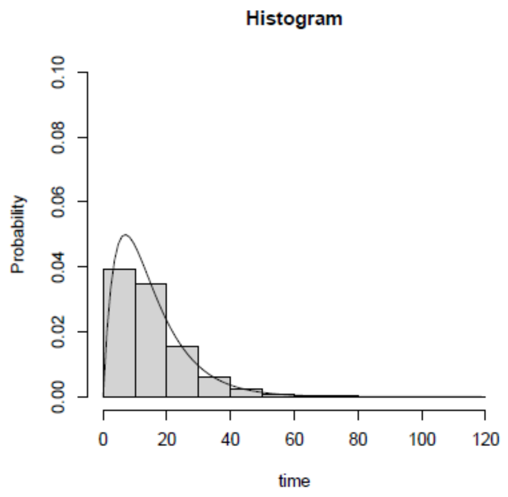
(b)

The figure below represents the histogram of 1000 draws of 'T' super-imposed with PDF $f_T(t)$. I have observed the similarities between the histogram and density as expected. The seven Monte Carlo estimates for E(T) and P(T > 15) based on 10000 draws and the corresponding solutions are shown in the table below. 1000 and 100000 based estimates are also represented in the table.

Table:

			Replication Number						
Parameter	True value	# Draws	1	2	3	4	5	6	7
E(T)	15	1000	15.884	14.764	15.146	15.157	15.250	15.492	14.963
E(T)	15	10000	14.981	15.001	15.018	15.103	14.875	15.100	15.151
E(T)	15	100000	15.016	14.969	15.049	15.016	15.017	14.990	15.003
P(T>15)	0.396473	1000	0.426	0.384	0.397	0.402	0.397	0.429	0.402
P(T>15)	0.396473	10000	0.396	0.395	0.399	0.402	0.392	0.400	0.404
P(T>15)	0.396473	100000	0.398	0.395	0.399	0.396	0.397	0.397	0.397

Figure:



R Program:

```
> set.seed(243)
> x <- replicate(10000, max(rexp(1, (1/10)), rexp(1, (1/10))))
> hist(x, prob = T, ylim = c(0, 0.1), xlab = "time", ylab = "Probability",
main = "Histogram
")
> pdf.f <- function(x){
+ return(0.2*exp(-0.1*x) - 0.2*exp(-0.2*x))
+ }
> curve(pdf.f, from = 0, to = max(x), add = T)
> simulation.f <- function(nsim, lambda.A = (1/10), lambda.B = (1/10)){
+ x <- replicate(nsim, max(rexp(1, lambda.A), rexp(1, lambda.B)))
+ result <- c(mean = mean(x), prob = mean(x > 15))
+ return(result)
+ }
> set.seed(243)
>

> round(replicate(7, simulation.f(1000)), 3)
[,1] [,2] [,3] [,4] [,5] [,6] [,7]
mean 15.884 14.764 15.146 15.157 15.250 15.492 14.963
prob 0.426 0.384 0.397 0.402 0.397 0.429 0.402

> round(replicate(7, simulation.f(10000)), 3)
[,1] [,2] [,3] [,4] [,5] [,6] [,7]
mean 14.981 15.001 15.018 15.103 14.875 15.1 15.151
prob 0.396 0.395 0.399 0.402 0.392 0.4 0.404

> round(replicate(7, simulation.f(100000)), 3)
[,1] [,2] [,3] [,4] [,5] [,6] [,7]
mean 15.016 14.969 15.049 15.016 15.017 14.990 15.003
prob 0.398 0.395 0.399 0.396 0.397 0.397 0.397
```

(c)

From LLN, approximations are close to true value. It is expected that the approximation accuracy improves as the # of draws increases. We repeat the process until a desired approximation is observed. After certain draws the approximations stall and will not improve the accuracy. As we can observe in the table, the results are consistent. The five approximations of E(T) and P(T>15) are almost identical to the true value. Variability in the approximation is decreased from true value when draws are increased from 1000 to 10000. The increase in approximation accuracy is less noticeable if the draws are increased from 10000 to 100000.

2.

In the hint it was given that unit square is inscribed in the circle. So A be the event which selects a random point of the square is inside the circle.

P(A) = (area of circle)/ (area of square) = Pi/4.

i.e., Pi = 4P(A). we can use the approximate of p(A) using Monte Carlo which is multiplied by 4 gives the approximate value of Pi. It generates a random point in the square. By simulating x and y coordinates of the point as an independent draw from a Uniform (0,1) distribution. The point (x,y) will be in the circle if its less than or equal to the radius of circle.

sqrt{$(x-0.5)^2 + (y-0.5)^2$} <= 0.5

to approximate the P(A), we need to simulate 10000 points in the square and apply the above equation to find if the point is inside the circle. Using R, P(A) is approximated to 0.7894.

Pi = 4 x (0.7894) = 3.1576

The above value when compared to 3.1416, which is the true value of Pi, it is approximately equal. With increase in the simulation to a greater value can increase the accuracy of the approximation.

R Program:

```
> pi.fun <- function(nsim){
+ x <- runif(nsim)
+ y <- runif(nsim)
+ prob <- mean(sqrt((x-0.5)^2 + (y-0.5)^2) <= 0.5)
+ pi <- 4*prob
+ return(c(prob = prob, pi = pi))
+ }

> set.seed(123)
> pi.fun(10000)
prob pi
0.7894 3.1576
> pi
[1] 3.141593
```